Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2005-31

ARC ROUTING PROBLEMS WITH TIME-DEPENDENT SERVICE COSTS

by

Mariam Tagmouti
Michel Gendreau
Jean-Yves Potvin

September 2005

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone : (514) 343-7575 / Télécopieur : (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

PUBLICATION
CRT-2005-31

ARC ROUTING PROBLEMS WITH TIME-DEPENDENT SERVICE COSTS

by

Mariam Tagmouti[1]
Michel Gendreau[1]
Jean-Yves Potvin[1]

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT).

[1] Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (mariamt;michelg;potvin@crt.umontreal.ca)

Département d'informatique et de recherche opérationnelle – Publication #1266

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2005

Résumé. Cet article étudie un problème de tournées sur les arcs avec contraintes de capacité et des coûts dépendant du temps de début de service. Ce problème s'inspire du problème de déglaçage des routes où le moment de chaque intervention est crucial. La méthode exacte de résolution présentée ici commence par transformer le problème de tournées sur les arcs en un problème de tournées sur les noeuds. Ensuite, une approche de génération de colonnes est utilisée pour résoudre ce dernier. Le problème maître est un problème classique de recouvrement, alors que les sous-problèmes sont des problèmes de plus court chemin élémentaire, dépendant du temps, avec contraintes de ressources. Ces sous-problèmes sont résolus en utilisant une extension d'un algorithme précédemment développé. Des résultats numériques sont présentés sur des problèmes dérivés d'un ensemble d'instances classiques pour le problème de tournées de véhicule avec fenêtres de temps.

Mots-clés tournées sur les arcs, génération de colonnes, chemin le plus court élémentaire, contraintes de ressources.

Abstract. This paper studies an arc routing problem with capacity constraints and time-dependent service costs. This problem is motivated by winter gritting applications where the "timing" of each intervention is crucial. The exact problem-solving approach reported here first transforms the arc routing problem into an equivalent node routing problem. Then, a column generation scheme is used to solve the latter. The master problem is a classical set covering problem, while the subproblems are elementary time-dependent shortest path problems with resource constraints. These subproblems are solved using an extension of a previously developed algorithm. Computational results are reported on problems derived from a set of classical instances of the vehicle routing problem with time windows.

Keywords arc routing, column generation, elementary shortest paths, resource constraints.

105-07480/

1 Introduction

We study a variant of the Capacitated Arc Routing Problem (CARP), introduced in [14], where a subset of arcs must be serviced at a cost that depends on the time of beginning of service. In this paper, the cost is a piecewise linear function of time. This problem is motivated from winter gritting operations, where the timing of an intervention is of prime importance [2, 8, 9, 20, 21]. That is, if the intervention is too early or too late, the cost in material and time sharply increases.

The CARP is NP-hard and was first addressed with relatively simple heuristics, like the construct-strike [3], path-scanning [15] and augment-merge [14] heuristics. They have been improved over time, mainly through the use of metaheuristics, like tabu search [16] or genetic algorithms [19]. An interesting variant of the CARP is presented in [13], where intermediate facilities are available to reload the vehicle. Although we do not address this variant here, these intermediate facilities could correspond to salt boxes, for example, in winter gritting applications. Surveys on the CARP and some of its variants can be found in [1, 5, 7, 10, 11]. We are not aware of any variant of the CARP with time-dependent service costs, although a particular case would be the CARP with soft time windows, when the service cost is "flat" within a given time interval and then increases linearly on both sides of the interval.

The paper is organized as follows. The problem is first introduced in Section 2. Then, its transformation into an equivalent node routing problem is described and the resulting mathematical formulation is presented in Section 3. The column generation approach for solving this problem is reported in Section 4, where time-dependent shortest path subproblems with resource constraints are addressed by extending a previously reported algorithm. In Section 5, numerical results on problems derived from instances of the vehicle routing problem with time windows (VRPTWs) are reported [22].

2 Problem description

Let $G = (V,A)$ be a directed graph where V is the vertex set and A is the arc set. We assume that A is partitioned into a subset of required arcs A_1, which must be serviced, and a complementary subset of arcs A_2. With each required arc $e \in A_1$ is associated a demand d_e, a length l_e, a travel time tt_e, a service time st_e, a travel cost tc_e and a time-dependent piecewise linear service cost function $sc_e(T_e)$, where T_e is the time of beginning of service on arc e. The other arcs in subset A_2 have a length, a travel time and a travel cost only. Note that the service time is typically larger than the travel time because it takes more time to service an arc than to simply travel along the arc.

A set $K = \{1,...,m\}$ of identical vehicles with capacity Q is available to service the required arcs. These vehicles are located at a central depot node from which each vehicle services a single route that starts and ends at the depot. The vehicles are not allowed to wait along their route and must be back at the depot by a given deadline. The objective is to service all required arcs in the graph at least cost with feasible routes,

where the cost is related to the number of vehicles used, the travel cost and the service cost.

Figure 1 shows typical piecewise linear service cost functions for the required arcs. Note that the fuction in Figure 1a is a degenerate form of the one shown in Figure 1b, where the "optimal" time interval for service reduces to a single point. The function shown in Figure 1b was used in our computational experiments, due to its similarity with classical "soft" time windows, but other piecewise linear forms could have been used as well, like the one shown in Figure 1c.

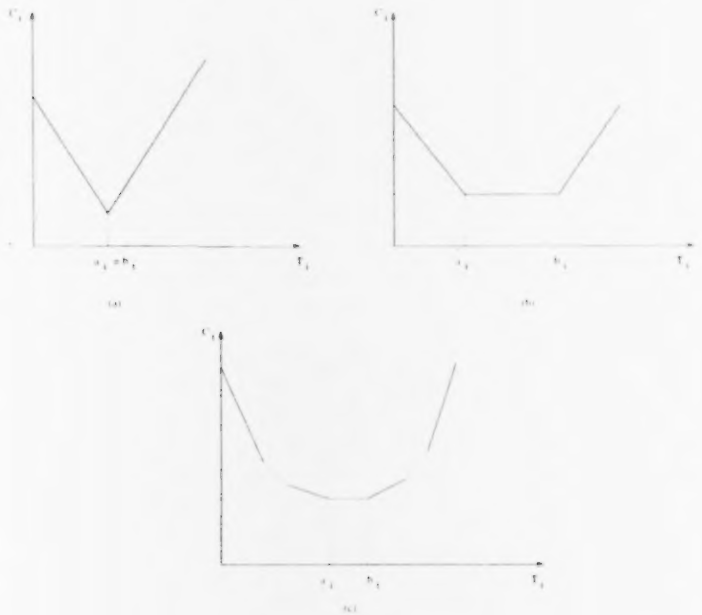

Figure 1: Different service cost functions

3 Problem Formulation

The first step is to transform the arc routing problem in graph $G = (V,A)$ into an equivalent node routing problem in a transformed graph $G' = (V',A')$. That is, each required arc $e_i \in A_1$ corresponds to a node i in graph G' with demand d_i, service time st_i and time-dependent service cost $sc_i(T_i)$. Each pair of distinct nodes i and j in G' is con-

nected by an arc $(i, j) \in A'$ with length l_{ij}, travel time tt_{ij} and travel cost tc_{ij}. The latter values are those of the shortest path between the two corresponding required arcs in graph G. These shortest paths are calculated from the end node of the first required arc to the end node of the second required arc. Finally, the central depot node is added and connected to all other nodes in G'. The arc values in the latter case are given by the shortest paths from the depot node to the end node of all required arcs in G or from the end node of all required arcs to the depot node, depending on the orientation of the arc.

Figure 2 illustrates this transformation. In the figure, the solid arcs in G are required arcs while dotted arcs are non required ones. Nodes i and j in G' correspond to arcs e_i and e_j in G, respectively. Each arc $e \in G$ is labeled with the vector $(d_e, st_e, l_e, tt_e, tc_e)$. Nodes $i, j \in G'$ are labeled with vectors (d_{e_i}, st_{e_i}) and (d_{e_j}, st_{e_j}), respectively, while arc $(i, j) \in G'$ is labeled with the vector $(l_{ij}, tt_{ij}, tc_{ij})$.

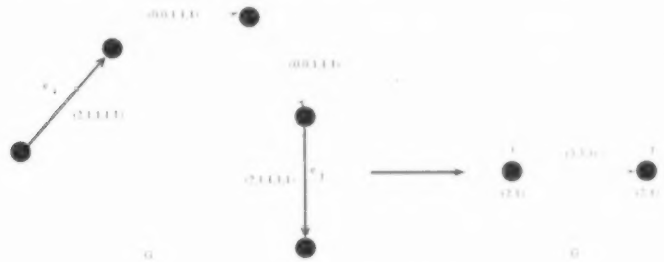

Figure 2: Graph transformation

After the transformation, we obtain a vehicle routing problem with time-dependent service costs. This type of problem has never been addressed in the literature, although some variants of the VRPTW can be related to it (see [17, 18, 23], for example, where missing a time window or a desired service time within a time window is penalized). A good review of different time-constrained vehicle routing problems can also be found in [6].

In the problem formulation below, N' stands for the set of nodes V' minus the depot (i.e., the nodes that must be served). Also, the depot is duplicated into an origin depot o and a destination depot d in V'. The decision variables are: (1) the binary flow variables on the arcs $x_{ij}^k, (i, j) \in A', k \in K$, which are equal to 1 if vehicle k travels on arc (i, j) to service node j, 0 otherwise, (2) the non negative load variables $Q_i^k, i \in V'$ which specify the load of vehicle k just after serving node i and (3) the non negative time variables $T_i^k, i \in V'$, which specify the time of beginning of service of vehicle k at node i. Note that $Q_o^k = Q$, $k \in K$, $d_o = d_d = 0$ and that T_o^k and T_d^k stand for the departure and return time of vehicle k at the depot.

$$\text{Min} \sum_{k \in K} \left(\sum_{(i,j) \in A'} tc_{ij} x_{ij}^k + \sum_{i \in N'} sc_i(T_i^k) \sum_{j \in N' \cup \{a\}} x_{ji}^k \right)$$

Subject to

$$\sum_{k \in K} \sum_{i \in N' \cup \{a\}} x_{ij}^k = 1 \qquad j \in N' \qquad (1)$$

$$\sum_{k \in K} \sum_{j \in N'} x_{oj}^k \leq m \qquad (2)$$

$$\sum_{j \in N' \cup \{d\}} x_{oj}^k = 1 \qquad k \in K \qquad (3)$$

$$\sum_{j \in N' \cup \{d\}} x_{ij}^k - \sum_{j \in N' \cup \{a\}} x_{ji}^k = 0 \qquad k \in K \quad j \in N' \qquad (4)$$

$$\sum_{i \in N' \cup \{a\}} x_{id}^k = 1 \qquad k \in K \qquad (5)$$

$$x_{ij}^k (T_i^k + st_i + tt_{ij} - T_j^k) \leq 0 \qquad k \in K \quad (i,j) \in A' \qquad (6)$$

$$x_{ij}^k (Q_i^k - d_j - Q_j^k) \leq 0 \qquad k \in K \quad (i,j) \in A' \qquad (7)$$

$$0 \leq T_i^k \leq T \qquad k \in K \quad i \in V' \qquad (8)$$

$$0 \leq Q_i^k \leq Q \qquad k \in K \quad i \in V' \qquad (9)$$

$$0 \leq x_{ij}^k \leq 1 \qquad k \in K \quad (i,j) \in A' \qquad (10)$$

$$x_{ij}^k \in \{0,1\} \qquad k \in K \quad (i,j) \in A' \qquad (11)$$

The objective is to minimize the sum of travel costs and time-dependent service costs. A fixed charge can also be added to the travel costs $tc_{oi}, i \in N'$, if one wants to penalize the use of an additional vehicle. Constraints (1) require that each node in N' be serviced once. Constraints (2) impose an upper bound m on the number of vehicles. Constraints (3) (4) and (5) are the flow conservation constraints. Constraints (6) and (7) ensure the feasibility of the time schedule and loads, respectively. Constraints (8) impose that the time of beginning of service at each node (including the departure and return time at the depot) be a non negative value that does not exceed the deadline T. Constraints (9), (10) and (11) ask for non negative load values that do not exceed vehicle capacity Q and binary values for the flow variables. Note that contraints (6) and (7) can be linearized, due to the presence of binary flow variables (see [6], for details). In the next section, a Dantzig-Wolfe decomposition, or column generation, scheme is thus proposed to solve this problem.

4 Column Generation

The column generation scheme proposed here is well documented in the literature. Consequently, we will only introduce the master problem and the shortest path sub-problems. The interested reader will find more details about column generation in [6].

The master problem corresponds to constraints (1) and (2) in the original formulation and can be expressed as follows:

$$\text{Min} \sum_{p \in \Omega} C_p u_p$$

Subject to

$$\sum_{p \in \Omega} a_{ip} u_p = 1 \qquad i \in N'$$

$$\sum_{p \in \Omega} u_p \leq m$$

$$u_p \geq 0 \qquad p \in \Omega$$

where decision variable u_p is 1 if path p is selected, 0 otherwise. In this formulation, Ω is the set of all feasible paths from the origin depot o to the destination depot d, C_p is the total cost of path p (sum of travel costs and service costs on all arcs and nodes along the path), and a_{ip} is 1 if node i is in path p. It is thus a linear relaxation of a set covering problem with an additional constraint on the total number of vehicles.

The subproblem, for each vehicle k, is an elementary shortest (least cost) path problem with resource constraints and corresponds to constraints (3) to (11) in the original formulation.

The resource constraints are the capacity constraint and the time deadline for the return of the vehicle at the depot. Denoting the reduced travel cost on arc $(i, j) \in A'$ by $\bar{tc}_{ij}$, the subproblem for a given vehicle k is expressed as follows:

$$\text{Min} \sum_{(i,j)\in A'} \bar{tc}_{ij} x_{ij}^k + \sum_{i\in N'} sc_i(T_i^k) \sum_{j\in N'\cup\{o\}} x_{ji}^k$$
Subject to
(3) to (11).

We start with an initial set of columns (paths) in the master problem, using a solution construction approach based on the savings heuristic of Clarke and Wright [4]. This simple heuristic works as follows. At the start, it is assumed that each node is serviced by a single route. Then, at each iteration, a pair of routes is selected and merged together on the basis of the best cost saving that can be achieved. This is repeated until a single route is obtained or no feasible merge exits. It is worth noting that the evaluation of the savings is based on the true time-dependent costs, as it is explained in Section 5.

The dual variables associated with this initial set of columns are then used to calculate the reduced travel costs in the corresponding subproblem. The latter is an elementary shortest path problem with resource constraints, which is solved with the algorithm of Feillet et al. [12]. All non dominated paths of negative cost are then added to the master problem and the latter is solved with CPLEX to obtain new dual variables. This procedure is repeated until no more paths with negative cost are found. Note that we incorporate as many paths as possible into the master problem, since the latter is relatively easy to solve when compared with the shortest path subproblems. Different filtering schemes have been tried to reduce the number of columns in the master problem, but these have invariably led to less efficient algorithms.

To obtain an integer solution, the column generation scheme is embedded within a previously reported branch-and-bound algorithm, where branching takes place on the arcs of the graph (i.e., an arc is forced into or excluded from a solution). More details about this algorithm can be found in [12].

The main difficulty when solving the shortest path subproblems stems from the time-dependent service costs that must be taken into account when evaluating a path. The algorithm of Feillet et al. was thus extended to address this issue. The original algorithm and the new extension are described in the following.

5 Elementary Shortest Path with Resource Constraints

The algorithm of Feillet et al. is a label correcting algorithm that solves the elementary shortest path problem with resource constraints on graphs with, possibly, negative cycles. In this context, a path is characterized by the consumption of each resource, in addition to its cost. When different paths lead to the same node, it might well be that no path dominates, or is better than the others, over all criteria. As a consequence, many different labels are typically maintained at each node (i.e., all non-dominated paths leading to that node).

Since elementary paths must be generated, cycles are detected by keeping a trace of previously visited nodes. More precisely, a path p from some origin node o to some node j is labeled with $R_p = (C_p, t_p^1, \ldots, t_p^l, s_p, V_p^1, \ldots, V_p^n)$, where $L = \{1, \ldots, l\}$ is the set of resources, C_p is the cost of path p, t_p^k is the consumption of resource $k = 1, \ldots, l$, s_p is the number of unreachable nodes (either because they have already been visited or because their inclusion would violate one or more resource constraints) and $V_p^i = 1$ if node i is unreachable, 0 otherwise. The following dominance relation is then defined:

Dominance relation. If p and p' are two different paths from origin o to node j with labels R_p and $R_{p'}$, respectively, then path p dominates p' if and only if $C_p \leq C_{p'}$, $s_p \leq s_{p'}$, $t_p^k \leq t_{p'}^k$, $k = 1, \ldots, l$, $V_p^i \leq V_{p'}^i$, $i = 1, \ldots, n$.

That is, path p dominates p' if (1) it is not more costly, (2) it does not consume more resources for every resource considered and (3) every unreachable node is also unreachable for path p'. Note that s_p, the number of unreachable nodes, is included in the label only to speed up the computations. As stated in [12], by eliminating paths through this dominance relation, only labels corresponding to non-dominated elementary paths are kept and a solution to the problem is obtained at the end.

In our application, there are $l = 2$ resource constraints: the capacity constraint (where the resources consumed are load units) and the deadline constraint for the return of the vehicle at the depot (where the resource consumed is time). As the algorithm is executed, we maintain and update at each node all non dominated shortest paths leading from the origin depot to that node. In particular, with each path p is associated its time-dependent total cost $C_p(T_0^p)$ *which is expressed as a function of departure time variable* T_0^p. Given some path p from the origin depot to node i with cost function $C_p(T_0^p)$, the cost function of the new path p' obtained by extending path p from node i to node j is simply:

$$C_{p'}(T_0^p) = C_p(T_0^p) + sc_j(T_0^p + TT_i + tt_{ij}) + \tilde{tc}_{ij},$$

where TT_i is the total travel time from the origin depot to node i.

An example is provided for a complete path in the Euclidean plane from the origin depot o to the destination depot d (see Figure 3). It is assumed that the travel time and the travel cost are the same and correspond to the Euclidean distance. The latter values are 2.8, 2.2, 3.6, 1.4, and 2.0 for arcs (o,1), (1,2), (2,3), (3,4) and (4,d), respectively.

The time-dependent service cost function for each vertex is also shown in Figure 4. The total cost function associated with the destination depot d is given in Figure 5 (where only the portion in the positive quadrant is considered). The minimum cost of this path is thus 24.8 for any T_0 between 0.4 and 2.

As we associate a cost function (rather than a cost value) with each path, the dominance relation between two partial paths that lead to the same node holds only when one function is "under" the other in the positive quadrant, as illustrated in Figure 6.

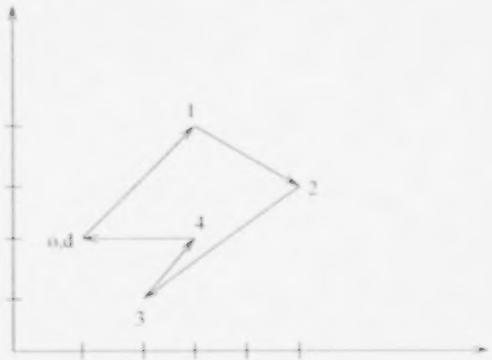

Figure 3: A complete path

It is worth noting that these cost functions can be easily handled within the Clarke and Wright heuristic, which is used to generate the initial set of columns. This is due to the merging process which consists in appending one path at the end of another. Let $p1$ and $p2$ be two paths from the origin depot o to the destination depot d with cost functions $C_{p1}\left(T_0^{p1}\right)$ and $C_{p2}\left(T_0^{p2}\right)$, where T_0^{p1} and T_0^{p2} denote the departure time variables associated with paths $p1$ and $p2$, respectively. Let also l and f be the last node in path $p1$ (just before destination depot d) and the first node in path $p2$ (just after the origin depot o), respectively. Then, if we merge $p1$ and $p2$ to form the new

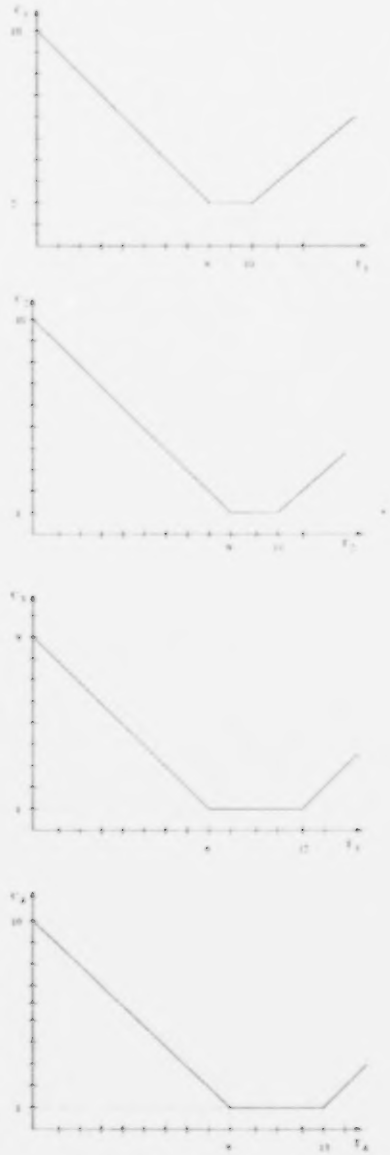

Figure 4: Service cost for each vertex $i = 1,2,3,4$

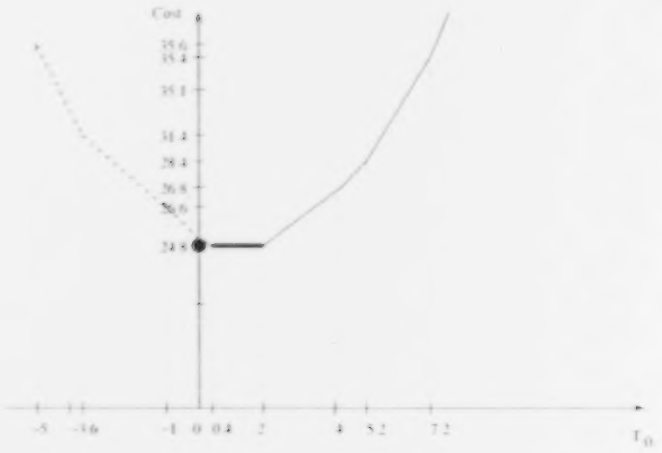

Figure 5: Total cost

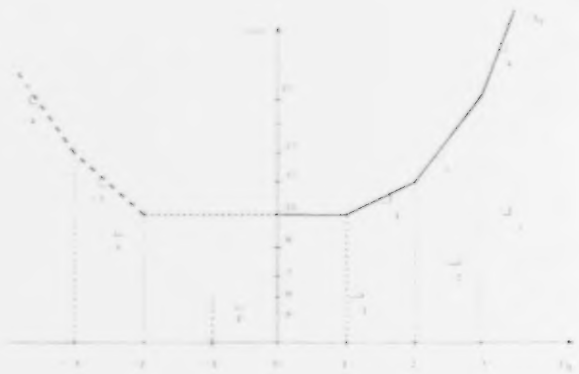

Figure 6: Example of dominance relation between two cost functions

path p, we can easily express its cost C_p in function of a single departure time variable. Namely, we have:

$$T_0^{p2} = T_0^{p1} + TT_l + tt_{lf} - tt_{of}$$

and we then make the replacement in $C_{p2}\left(T_0^{p2}\right)$ to obtain C_{p2} in function of T_0^{p1}. Then, the total cost of path p can be expressed in function of the single departure time variable T_0^{p1} as follows:

$$C_p\left(T_0^{p1}\right) = C_{p1}\left(T_0^{p1}\right) - tc_{ld} + C_{p2}\left(T_0^{p1}\right) - tc_{of} + tc_{lf}.$$

It is thus an easy matter to update the cost functions as the Clarke and Wright heuristic unfolds.

In the following, numerical results obtained with our column generation algorithm are reported.

6 Numerical Results

For numerical testing, Solomon's VRPTW benchmark problems were used [22]. We have focused on problem instances of type 1 which are easier to solve with our exact algorithm, due to a restrictive time deadline that leads to shorter routes. As these problems are Euclidean, the length, travel time and travel cost between two nodes are identical and correspond to the Euclidean distance. The service cost function at each node was specified as follows. First, the "flat" or minimum cost interval of the curve corresponds to the original time window in Solomon's instances. Second, the minimum cost was set to the service time at the node, as defined in Solomon's instances. Finally, a slope parameter was added to specify a linear cost increase on both sides of the window. The shape of the time-dependent service cost at each node is thus similar to the one shown in Figure 1b.

6.1 Parameter study

Our first results were obtained on the 25-customer instances in class R1 (random customer locations) and RC1 (mix of random and clustered customer locations). These results are shown in Table 1, with slopes of +1 and -1 on both sides of the time window and no penalty for the use of vehicles (recall that this penalty can be included in the travel cost from the depot to the first node on a route). In this Table, the first six columns provide the instance number, initial solution cost provided by the Clarke and Wright heuristic, optimal solution cost, number of vehicles used, solution length (i.e., total travel distance) and computation time in seconds. In the last two columns, we indicate the total number of columns or paths generated by our algorithm and the total number of calls to the shortest path algorithm, respectively. These tests were run on a 400 MHz UltraSpare II processor.

Table 2 shows the results obtained when the slope is set to -10000 and +10000 on both sides of the time window. In this case, the algorithm is "virtually" forced to visit each node within its window, otherwise a huge penalty is incurred. We thus obtain a solution to a vehicle routing problem with hard time windows, where "hard" applies to both the lower and upper bound of the window. This feasibility/infeasibility issue leads to the larger variance observed in the CPU times. For some instances, a feasible solution is quickly found, which allows the algorithm to prune the search space early. Conversely, for some other instances, it is much more difficult to identify a feasible solution. Also, it is clear that a larger number of vehicles is needed to meet the time windows in this case.

In Table 3 the slopes are reset to +1 and -1, but a penalty of 50 Euclidean units is associated with the use of each vehicle. As expected, the number of vehicles is reduced when compared with the numbers shown in Table 1.

Instance	Initial Cost	Optimal Cost	#Vehicles	Length	CPU Time	#Columns (sec)	#Iterations
R101	949.4	893.7	5	532.2	1485.6	1037	16
R102	861.2	806.0	5	479.2	2357.7	2565	41
R103	799.5	726.1	4	439.2	462.9	2339	11
R104	713.5	690.0	4	415.3	3283.8	5463	50
R105	959.3	780.4	5	487.3	679.8	1366	17
R106	767.8	721.4	4	454.2	1362.0	2847	45
R107	710.4	679.5	4	427.4	792.6	4062	28
R108	687.7	647.7	3	379.6	1338.9	8610	50
R109	714.5	691.3	5	441.3	412.4	1598	13
R110	696.7	668.8	4	407.6	724.0	3237	14
R111	712.9	676.3	4	416.8	907.6	2974	34
R112	703.7	643.0	4	393.0	2047.4	7673	50
RC101	844.1	623.8	3	362.4	358.8	1945	14
RC102	773.4	598.3	3	344.0	1034.1	2846	16
RC103	711.3	585.1	3	335.1	5413.4	5554	23
RC104	728.6	572.4	3	322.4	18302.8	8999	19
RC105	816.8	623.1	3	359.4	4276.3	3770	16
RC106	760	588.7	3	327.6	2142.3	2557	14
RC107	735.9	548.3	3	298.3	4742.4	2771	18
RC108	709.5	544.5	3	294.5	19878.1	5986	19

Table 1: Results on 25-customer instances with slopes +1 and -1

Instance	Initial Cost	Optimal Cost	#Vehicles	Length	CPU Time (sec)	#Columns	#Iterations
R101	1083.0	1068.6	11	818.6	72.1	83	7
R102	910.6	887	7	637.0	429.5	1241	50
R103	811.6	755	5	505.0	293.3	1356	28
R104	744.1	694.4	4	444.4	242.2	2267	15
R105	914.6	847.5	6	597.5	489.1	478	21
R106	800	752.2	6	502.2	611.1	1961	29
R107	754.4	683.9	4	433.9	203.9	2415	11
R108	723.1	660.3	4	410.3	1336.3	4086	50
R109	744.2	691.3	5	441.3	871.3	1423	12
R110	735.7	694.1	5	444.1	42437.5	2848	50
R111	734.4	684.7	4	434.7	836.3	1913	30
R112	691.1	643.0	4	393.0	2094.0	6365	50
RC101	945.7	724	4	474.0	291.5	702	50
RC102	866.3	601.8	3	351.8	269.7	1853	15
RC103	738.5	585.1	3	335.1	11418.4	3109	15
RC104	765.9	574.8	3	324.8	3737.7	3894	19
RC105	952	687.2	4	437.2	428.5	3770	16
RC106	747.2	595.5	3	345.5	1148.5	1528	17
RC107	717.2	548.3	3	298.3	1477.5	2766	16
RC108	650.6	544.5	3	294.5	4793.8	5803	15

Table 2: Results on 25-customer instances with slopes +10000 and -10000

Instance	Initial Cost	Optimal Cost	#Vehicles	Length	CPU Time (sec)	#Columns	#Iterations
R101	1249.4	1140.3	4	510.3	4153.2	3958	37
R102	1161.2	1016.5	4	483.2	28149.8	6839	50
R103	1249.4	926.1	4	439.2	70031.9	8956	50
R104	979.8	835.9	3	408.3	58008.3	13625	25
R105	1275.2	991.1	4	488.7	1683.7	3416	12
R106	1017.8	905.2	3	397.5	20287.7	5606	21
R107	963.3	874.6	3	410.3	109674.0	14741	50
R108	869.9	797.7	3	379.6	32309.9	14598	28
R109	955.1	904.3	4	442.4	5505.0	4488	20
R110	931.6	868.8	4	407.6	29219.3	7983	50
R111	952.4	875.7	3	426.1	61181.5	9558	50
R112	941.1	803.1	3	389.4	60551.4	19266	49
RC101	1166.3	773.8	3	362.4	1551.2	4560	17
RC102	1002.5	748.3	3	344.0	21090.2	7870	21
RC103	911.3	735.1	3	335.1	60319.4	10398	16
RC104	928.6	724.8	3	324.8	89256.0	12687	20
RC105	1110.3	773.1	3	359.4	7201.9	7846	15
RC106	1013.6	745.5	3	345.5	1748.2	2127	17
RC107	937.5	697.3	3	296.3	56334.7	9174	18
RC108	913.5	694.5	3	294.5	253741.0	12830	20

Table 3: Results on 25-customer instances with slopes +1 and -1 and vehicle penalty

6.2 Problem size

As our algorithm is an exact one, this section shows how "far" we can go with regard to problem size. To this end, we used a more powerful machine with a 2.4 GHz ADM Opteron(tm) 250 processor. In all those tests, the slopes were set to +1 and -1 and there was no penalty for the use of a vehicle. Also, the tests were restricted to the R1 problem class. As we can see in Tables 4 and 5, the limit is reached with only a slightly larger number of customers, even if a more powerful machine was used. More precisely, a sharp increase in computation times is observed from the instances with 25 customers to the instances with 35 customers (e.g., one day of computation for R110). When the number of customers grows to 40, only 6 instances out of 12 were solved to completion. For the remaining instances, the algorithm stopped due to insufficient memory.

7 Conclusion

In this paper, we have considered an arc routing problem with time-dependent service costs. To address this problem, we first transformed it into an equivalent node routing problem. The latter was solved exactly with a column generation approach. In the process, a previously reported elementary shortest path algorithm with resource constraints was extended to solve the time-dependent subproblems. Future research will now focus on problem formulations that are closer to the winter gritting application that motivated this work. Heuristic problem-solving approaches will then be considered.

Instance	Initial Cost	Cost	#Vehicles	Length	CPU Time (sec)	#Columns	#Iterations
R101	1216.9	1168.3	7	675.9	23953.6	3659	50
R102	1121.7	1048	7	608.1	37801.3	9978	50
R103	991.8	929.2	5	527.1	29496.8	13770	40
R104	885.8	837.8	4	469.8	35712.4	41532	70
R105	1045.1	1017.1	6	611.7	16825.6	5228	12
R106	1003.8	930.8	5	562.2	12921.1	14921	14
R107	928.5	874.8	5	514.6	51504.8	31736	70
R108	861.7	805.7	4	446.6	84516.5	96249	68
R109	953.9	914.8	6	559.7	46254.0	7708	48
R110	935.4	893.6	5	531.5	93493.0	11215	50
R111	942.5	864.4	5	493.9	57256.3	18455	40
R112	848.2	834.3	5	476.4	59672.9	28430	50

Table 4: Results on 35-customer instances

Instance	Initial Cost	Cost	#Vehicles	Length	CPU Time (sec)	#Columns	#Iterations
R101	1363.7	1318.4	8	778.7	25687.0	5953	28
R102	1280.4	1194.8	9	733.4	107191.0	15764	70
R103	—	—	–	—	—	—	–
R104	—	—	–	—	—	—	–
R105	1328.3	1171.2	7	721.7	21735.9	7127	22
R106	—	—	–	—	—	—	–
R107	1082.3	995.3	6	591.3	29696.3	34771	15
R108	—	—	–	—	—	—	–
R109	1318.6	1045.7	6	626.4	76125.3	12944	40
R110	—	—	–	—	—	—	–
R111	—	—	–	—	—	—	–
R112	999.9	937.1	5	529	165221.0	58859	50

Table 5: Results on 40-customer instances

Bibliography

[1] **Assad A.A. and B.L. Golden**, *Arc Routing Methods and Applications*, in: M. O. Ball et al. (eds), Network Routing, Elsevier, 375–483, 1995.

[2] **Campbell J.F. and A. Langevin**, *Roadway Snow and Ice Control*, in: M. Dror (ed.), Arc Routing: Theory, Solutions and Applications, Kluwer, 389–418, 2000.

[3] **Christofides N.**, *The Optimum Traversal of a Graph*, Omega 1, 719–732, 1973.

[4] **Clarke G. and J. Wright**, *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*, Operations Research 12, 568-581, 1964.

[5] **Cordeau J.F. and G. Laporte**, *Modeling and Optimization of Vehicle Routing and Arc Routing Problems*, Les Cahiers du GERAD G-2002-30, Montreal, 2002.

[6] **Desrosiers J., Y. Dumas, M.M. Solomon and F. Soumis**, *Time Constrained Routing and Scheduling*, in: M.O. Ball et al. (eds), Network Routing, Elsevier, 35–139, 1995.

[7] **Dror M.**, *Arc Routing: Theory, Solutions and Applications*, Kluwer, 2000.

[8] **Eglese R.W. and L.Y.O. Li**, *Efficient Routeing for Winter Gritting*, Journal of the Operational Research Society 43, 1031–1034, 1992.

[9] **Eglese R.W.**, *Routing Winter Gritting Vehicles*, Discrete Applied Mathematics 48, 231–244, 1994.

[10] **Eiselt H.A., M. Gendreau and G. Laporte**, *Arc Routing Problems, Part I: The Chinese Postman Problem*, Operations Research 43, 231–242, 1995.

[11] **Eiselt H.A., M. Gendreau and G. Laporte**, *Arc Routing Problems, Part II: The Rural Postman Problem*, Operations Research 43, 399–414, 1995.

[12] **Feillet D., P. Dejax, M. Gendreau and C. Gueguen**, *An Exact Algorithm for the Elementary Shortest Path Problem With Resource Constraints : Application to some Vehicle Routing Problems*, Networks 44, 216–229, 2004.

[13] **Ghiani G., G. Improta and G. Laporte** , *The Capacitated Arc Routing Problem with Intermediate Facilities*, Networks 37, 134–143, 2001.

[14] **Golden B.L. and R. Wong**, *Capacitated Arc Routing Problems*, Networks 11, 305–315, 1981.

[15] **Golden B.L., J.S. DeArmmon and E.K. Baker**, *Computational Experiments with Algorithms for a Class of Routing Problems*, Computers & Operations Research 10, 47–52, 1983.

[16] **Hertz A., G. Laporte and M. Mittaz**, *A Tabu Search Heuristic for the Capacitated Arc Routing Problem*, Operations Research 48, 129–135, 2000.

[17] **Ibaraki T., S. Imahori, M. Kubo, T. Masuda, T. Uno and M. Yagiura**, *Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints*, Transportation Science 39, 206–232, 2005.

[18] **Ioachim I., S. Gélinas, J. Desrosiers and F. Soumis**, *A Dynamic Programming Algorithm for the Shortest Path Problem With Time Windows and Linear Node Costs*, Networks 31, 193–204, 1998.

[19] **Lacomme P., C. Prins and W. Ramdane-Chérif**, *A GA for the CARP and its Extensions*, in: E.J.W. Boers et al. (eds), Applications of Evolutionary Computing, Lecture Notes in Computer Science 2037, Springer, 473–483, 2001.

[20] **Li L.Y.O. and R.W. Eglese**, *An Interactive Algorithm for Vehicle Routeing for Winter Gritting*, Journal of the Operational Research Society 47, 217–228, 1996.

[21] **Lotan T., D. Cattrysse and D. Van Oudheusden**, *Winter Gritting in the Province of Antwerp: a Combined Location and Routing Problem*, Technical Report IS-MG 96/4, Institut de Statistique, Université Libre de Bruxelles, 1996.

[22] **Solomon M.M.**, *Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints*, Operations Research 35, 254–265, 1987.

[23] **Taillard E.D., P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin**, *A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows*, Transportation Science 31, 170–186, 1997.